

3.3) 消息的confirm机制

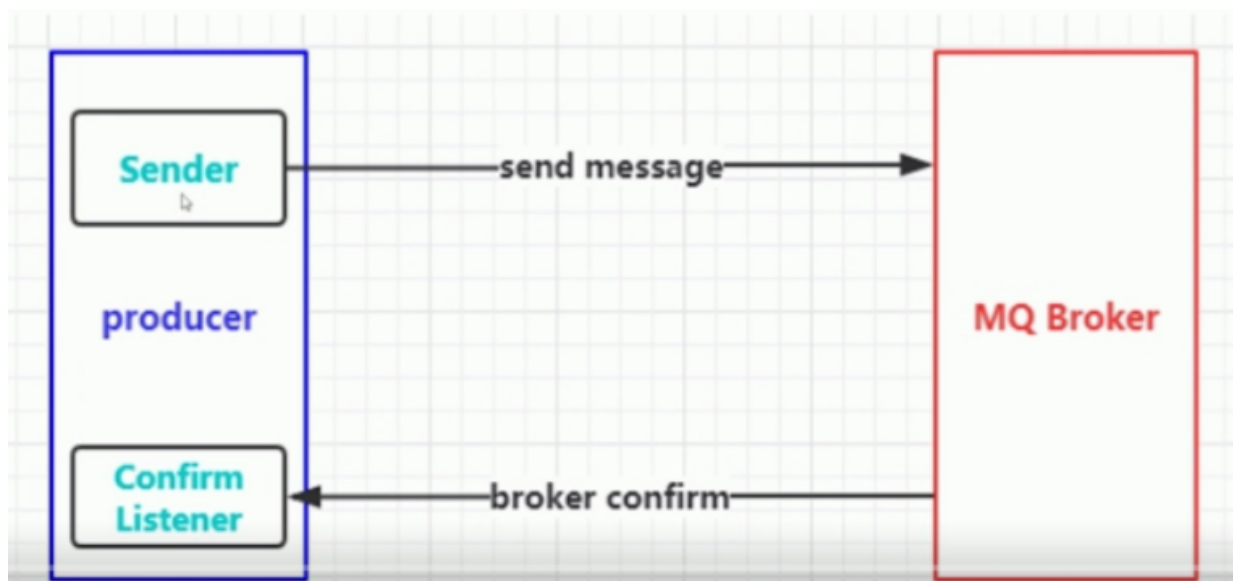
一:mq 的confirm机制

1: 消息的确认:指的是生产者将消息投递后, 如何mq-server接受到消息, 就会给生产者一个应答.

2:生产者接受到应答, 来确保该条消息是否成功发送到了mq-server

3:confirm机制是消息可靠性投递的核心保障

二:mq的confirm机制的核心流程图



三:confirm机制的现实步骤

第一步: 在channel 上开启确认模式 `channel.confirmSelect();`

第二步:在channel上添加监听, 用来监听mq-server返回的应答

四:代码演示

生产者:代码

```
public static void main(String[] args) throws IOException, TimeoutException {  
    ConnectionFactory connectionFactory = new ConnectionFactory();
```

```

connectionFactory.setVirtualHost("/");
connectionFactory.setHost("47.104.128.12");
connectionFactory.setPort(5672);
Connection connection = connectionFactory.newConnection();
Channel channel = connection.createChannel();
//开启confirm 确认机制
channel.confirmSelect();
//设置confirm 监听
channel.addConfirmListener(new AngleConfirmListerner());
//生产消息
channel.basicPublish("test.confirm.exchange","test.confirm.key",null,"测试confirm消息".getBytes());
}

```

消费者代码:

```

public static void main(String[] args) throws IOException, TimeoutException, InterruptedException {
ConnectionFactory connectionFactory = new ConnectionFactory();
connectionFactory.setVirtualHost("/");
connectionFactory.setHost("47.104.128.12");
connectionFactory.setPort(5672);
Connection connection = connectionFactory.newConnection();
Channel channel = connection.createChannel();
//声明交换机队列以及绑定关系
channel.exchangeDeclare("test.confirm.exchange","topic",true,true,false,null);
channel.queueDeclare("test.confirm.queue",true,false,true,null);
channel.queueBind("test.confirm.queue","test.confirm.exchange","test.confirm.key");
QueueingConsumer queueingConsumer = new QueueingConsumer(channel);
channel.basicConsume("test.confirm.queue",true,queueingConsumer);
while (true) {
QueueingConsumer.Delivery delivery = queueingConsumer.nextDelivery();
System.out.println(new String(delivery.getBody()));
}
}

```

confirm消息监听器代码

```

public class AngleConfirmListerner implements ConfirmListener {
@Override
public void handleAck(long deliveryTag, boolean multiple) throws IOException {
System.out.println("消息deliveryTag"+deliveryTag+"被正常签收");
}
@Override
public void handleNack(long deliveryTag, boolean multiple) throws IOException {
System.out.println("消息deliveryTag"+deliveryTag+"没被签收");
}
}

```

```
}  
}
```